Generic Face Animation

Mauricio Cerda, Renato Valenzuela, Nancy Hitschfeld-Kahler, Lucas Terissi, Juan C. Gomez

Generic Face Animation

Mauricio Cerda
INRIA-Loria Nancy Grand Est
Equipe Cortex - Bat. C040, 54506
Vandoeuvre-les-Nancy, France
Email: mauricio.cerda@loria.fr

Renato Valenzuela
Department of Computer Science,
Universidad de Chile,
Blanco Encalada 2120, Santiago, Chile
Email: revalenz@gmail.com

Nancy Hitschfeld-Kahler
Department of Computer Science,
Universidad de Chile,
Blanco Encalada 2120, Santiago, Chile
Email: nancy@dcc.uchile.cl

Lucas D. Terissi
Laboratory for System Dynamics and Signal Processing,
Universidad Nacional de Rosario, CIFASIS, CONICET,
Rosario, Argentina
Email: terissi@cifasis-conicet.gov.ar

Juan C. Gómez
Laboratory for System Dynamics and Signal Processing,
Universidad Nacional de Rosario, CIFASIS, CONICET,
Rosario, Argentina
Email: gomez@cifasis-conicet.gov.ar

***Abstract*—In computer vision, the animation of objects has attracted a lot attention, specially the animations of *3D* face models. The animation of face models requires in general to manually adapt each generic movement (open/close mouth) to each specific head geometry. In this work we propose a technique for the animation of any face model avoiding most of the manual intervention. In order to achieve this we assume that: (1) faces, despite obvious differences are quite similar and a single generic model can be used to simplify deformations and (2) using this face model, a simple interpolation technique can be used, with minimal manual intervention. Several examples are presented to verify the realism of the obtained animations.**

***Keywords*-computer graphics; animation; mesh deformation**

I. Introduction

In Computer Graphics, one important area of research is object deformation in particular *3D* animation. Computer animation is a large topic; with applications from artistic deformation usually found in the cinema, going through the visualization of physical simulations to aspects such as video compression and automatic video generation. In this field objects are very often modeled as *3D* meshes, where a deformation consists in the change of the position of each vertex defining the object. This work specifically deals with face animation under the perspective of how we can automatize the processes, still achieving realistic animations.

Facial animation in general aims to deform a face mesh (in *3D*) as realistic as possible. Face models are specified as at least one *3D* mesh, usually with texture information (for each *3D* point there is an associated *2D* point in a picture), see Figure 1(a). More complex models also include anatomic considerations (muscles, tendons, bones) [1]. A "normal" quality head model will typically have around 10^4 vertices, and it will be composed by several unconnected sub-meshes such as eyes, teeth's and the face it self.

To deform a face mesh, two widely used techniques are bone deformation and pose decomposition [1]. Bone deformation (an other related techniques) associates the *3D* head with a given internal structure, most often this structure mimic the main head bones. The skeleton deforms the external mesh as each vertex follows its closest bone. In this kind of technique the number of modeled bones is related with the available animation data (manual deformation, few or multiple tracking marks). Additionally, we must mention super-realistic animations [2] where the face is no longer a textured mesh but a set of layers disposed to mimic bones, skin and even hair; this last kind of animation is not considered in this work because it is not possible to our knowledge to obtain the animation data only by video tracking or audio, and a large amount of artistic and materials simulations work must be performed.

Pose decomposition in the other hand defines static deformation poses (position for each vertex) such as "smiling", "angry", "sad", "mouth open", "eyes closed". The main idea is that considering a large enough set of poses, any animation can be defined as combinations of poses with different weights along time [3]. The animation technique we propose in this work is more related to pose decomposition, in the sense that the generic face model we use delivers a set of surface points that we use in order to deform a high detailed mesh, directly deforming the mesh surface.

Our work is organized as follow; in the next section we describes mesh deformation techniques in general, and we introduce the problem that specifically motivates this work. In section III our proposed method is presented. The last two sections present some results and conclusions about our work.

II. Mesh deformation

Mesh deformation in general looks for the best vertex translation with the location of all vertices or most often, a subset of them. The best vertex translation can sometimes

be related with physical constraints [4], [5], or with the subjective perception of the final animation.

In the literature, mesh deformation has been developed since several years ago [6], [7], [8], [9] and we can find at least two different approaches. The first is to manually assign for each vertex the control points that will define its position, and to average in function of the relative distance between each control point and the target vertex [6], [7]. The second approach that we find usually in physical simulations it also considers a subset of control points to change the position of each vertex, but it tries to guarantee some property about the resulting deformation in the high detailed mesh [8], [9]. This last approach defines properties that are related to what the mesh represents [8], [9]. For example in aeroelastic applications to strictly follow control points and to respect rigid deformations (rotations and translations) ensures that the total load and momentum are conserved when going from the low to the high detailed mesh. Face Animation does not provide in general a way to define which properties must be followed to obtain a realistic result. Yet, in our work we explore an approach which guarantees a set of properties that we link with realistic animations.

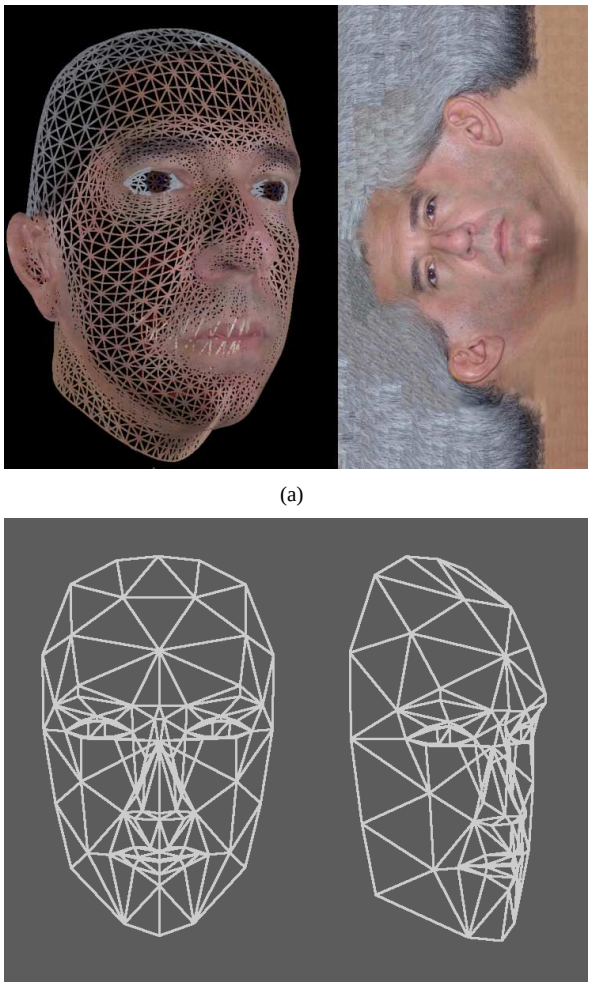
The manual technique we mention earlier in this section where the authors manually assign and weight each vertex with respect to the control points, represents a very tedious, time consuming, and imprecise method that very often reach poor results [6]. In the other hand, mesh deformation with constraints requires to give a precise definition of what we call realistic animation. In this work we propose an automatic algorithm that combines aspects of both techniques.

High detailed face mesh (10^4 vertices) deformation with a few control points (10^2) is the problem that motivates this work. We must mention that this specific problem has been addressed before, moreover, control points (feature points) have been already standardized in the "MPEG-4" facial deformation standard [10]. The fact to have standardized feature points makes generic animation even more suitable and encourage our work, as a more general solution.

III. Proposed method

Starting with the idea that we can generate or capture a few points of the face deformation, by means of video tracking [11] or audio analysis [12], we deal directly with the deformation of high detailed meshes. Any of these methods provide *3D* information of around 150 relevant points in the face: the control points, and we require to deform the complete *3D* head mesh, see Figure 1(a), using this information.

To simplify our problem we consider a single generic face model, the Candide3 model [13], a *3D* triangular mesh. This model provides a simplified face deformation model with more detail in areas where commonly higher deformations are observed, see Figure 1(b). The vertices of the Candide3 model correspond in general to the feature points defined



(a)

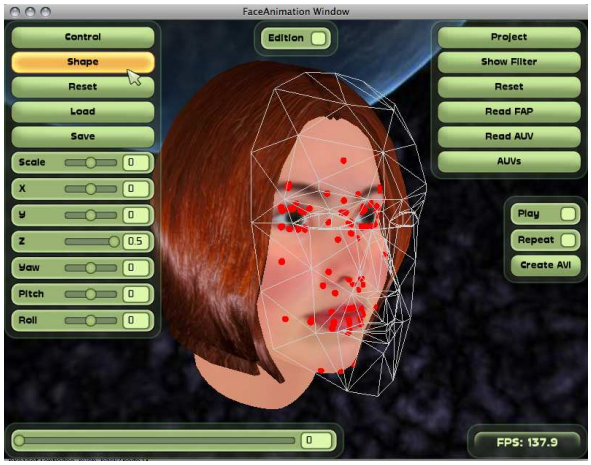
(b)

Figure 1. (a) The JuanCarlos *wrml* model (6000 vertices) and its texture. (b) The Candide3 model (156 vertices).

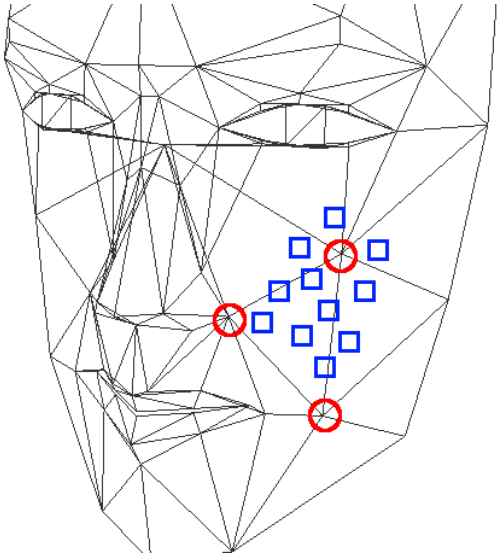
in the MPEG-4 standard [10], [13] and each action unit in MPEG-4 has an equivalence in the Candide3 Model. The main difference is that MPEG-4 only specify the vertices and not the mesh, and as we will see in this section, the mesh provides helpful information to perform the interpolation.

A. Adaptation of the Candide3 model

The first step in the generic animation process is to adjust the Candide3 model to match the form of the *3D* head model we want to deform. This implies not only to change size and position of the model but also to adjust anatomic parameters (distance between eyes and distance from eyes to mouth , among others). Despite the number of parameters, this step can be achieved in a few minutes, and a rough matching is sufficient; mouth and eyes position are at least necessary to match, see Figure 2(a). In our software we implement the matching with a few sliders. We decide not to automatize this process as faces can be used to deform arbitrary *3D*



(a)



(b)

Figure 2. a) Candide3 mesh projected (red dots) into the Alice wrml model. b) Schematic view of the projection (red circles) and the rest of the vertices (blue squares) that must be interpolated from them.

objects, where human intervention will be anyway required.

Once the Candide3 model is deformed to match the *3D* head mesh, a projection using the *3D* Euclidean distance is performed. After this second step, we will have one (an only one) vertex in the *3D* head model that corresponds to each Candide3 vertex. These special points are the control points in the *3D* head mesh that will later deform the complete mesh.

B. Interpolation with control points

After the projection of control points on the head mesh, we assume that:

- Control points must be followed in the *3D* head model.
- Under rigid deformation of 3 control points forming a face in the Candide3 model, each interpolated vertex should be deformed on the same way.
- Interpolation must be linear with respect to the control points for fast calculations.

We impose these three restrictions to build our interpolation technique. The first restriction stands that control points must be projected to only one head mesh point. The second implies that under the same translation or rotation of a set of control points, all related head mesh points must exhibit the same rotation and translation. The last restriction ensures that our animation technique will be fast.

In the *3D* head model we already know how to move a few points, the control points, as they are a direct projection from the Candide3 model, see Figure 2(a) or Figure 2(b) both in red. To move the rest of the points (third step), we associate each vertex to a set of control points. In order to do this, we first associate each head mesh vertex to a triangle of the Candide3 mesh, see Figure 2(b) with □. However, as it is defined in *3D*, we use the *2D* projection that provides the available texture. The method we propose is to use the 3 control points that define the triangle of the Candide mesh to define the interpolation as,

$$\vec{v}_k = \mu_0\vec{c}_{i-1} + \mu_1\vec{c}_i + \mu_2\vec{c}_{i+1} \tag{1}$$

where $\vec{v}_k$ is the initial position for each face mesh vertex, $\vec{c}$ is a control point, and $\vec{\mu} = (\mu_0, \mu_1, \mu_2)$ the 3 weights we want to calculate. The solution for the linear system is:

$$\vec{\mu} = C^{-1}\vec{v}_k \tag{2}$$

here C is the column matrix with the control points coordinates. Knowing $\vec{\mu}$ we can obtain by interpolation the position of any head mesh vertex coordinate by weighting the control points positions and $\vec{\mu}$, see Eq. 1. We will now show the properties of this interpolation method.

- *Control points*. Suppose $\vec{v}_k = \vec{c}_{i-1}$ (for any of the 3 control points), *i.e.* we try to interpolate one of the control points, then the solution of $\vec{\mu}$ is:

$$\vec{0} = C\begin{pmatrix} 1-\mu_0 \\ \mu_1 \\ \mu_2 \end{pmatrix} \tag{3}$$

  as C is not null, the solution implies that $\mu_0 = 1$ and $\mu_1 = 0,\ \mu_2 = 0$. Then this interpolation considers the first constraint: control points are always followed.
- *Deformations*. There are two possible rigid deformations rotations (R), and translations (T). Depending on the linear model we use, we can make the method to respect either rotations or translations. Suppose a rotation over the control points, then the new coordinate of any vertex will be defined as:

$$\vec{v}'_k = \mu_0 R\vec{c}_{i-1} + \mu_1 R\vec{c}_i + \mu_2 R\vec{c}_{i+1} = R\vec{v}_k \tag{4}$$

  in this equation we apply the same rotation to each control point. We see that $\vec{v}'_k = R\vec{v}_k$, then under rotation the

interpolation respect the solid deformation. Following the same idea, we can guarantee to follow translations but we need to further impose that $\Sigma\mu_i = 1$, therefore to add a constant to the interpolation.

- *Linearity*. By construction the interpolation is linear.

C. Implementation

There are two main implementation issues in the method we propose. In the adaptation stage of the Candide3 model, is very common to break the topology with the projection, specially in areas such as the lips because the *3D* Euclidean distance does not avoid that mesh vertices of the Candide3 superior lip can be projected into the inferior lips of the head mesh. The second issue is the case when a *3D* head vertex is not inside any triangle in the projected texture, *i.e.* Eq. 4 is no longer valid. This happens usually in the interior part of the lip and in the back part of the head. Both problems are now explained more precisely with the algorithms we use to overcome them.

1) Topology preservation: As we mention, in the adaptation of the Candide3 mesh to the *3D* head mesh, is common to break the topology. In order to handle this we verify the geodesic distance [14] between each of each projected vertex, to check that those forming a triangle in the Candide3 mesh are close in the head mesh. This verification must be carried on for each Candide3 triangle: if in one of the three vertices geodesic distance to the other two, the euclidean distance is not strictly reduced at each step, *i.e.* there is a "jump" between two vertices, the next closest vertex is selected with the 3D Euclidean distance.

2) Border Handling: Our proposed method requires to associate for each head mesh vertex a certain Candide3 face. As all vertices are *3D*, we define the notion of inside a certain Candide triangle as in *2D* using the available texture information. This solution has the inconvenient to generate points that are outside all Candide3 triangles. To handle these cases we use the interpolation,

$$\vec{v}_k = \vec{c}_{i-1} e^{-\frac{|\vec{v}_k - c_{i-1}|}{\alpha^2}} + \vec{c}_i e^{-\frac{|\vec{v}_k - c_i|}{\alpha^2}} + \vec{c}_{i+1} e^{-\frac{|\vec{v}_k - c_{i+1}|}{\alpha^2}} \quad (5)$$

where α is manually selected and fixed for all head mesh vertices and $\vec{c}_{i-1}$, $\vec{c}_i$ and $\vec{c}_{i+1}$ are the vertices of the closest directly connected Candide triangle. We define the distance from a vertex to a triangle as the sum of the 0 distances from the vertex to each triangle vertex.

3) Algorithm order: The proposed algorithm is composed of two stages: adaptation and interpolation. One important aspect of the method we have presented is that a linear interpolation model allows for fast calculations. Assuming that the Candide3 mesh has *n* vertices and the *3D* head mesh has *N* vertices, and that we know the interpolation vector $\vec{\mu}$, any face deformation requires $O(3N)$ operations (points are in *3D*). The computation of $\vec{\mu}$ for each vertex is more expensive but it is performed only once. For μ, the deformation process computes the closest vertex in $O(Nn)$ and then it must check the topology. In the worst case this may imply again $O(Nn)$ operations. After the projection step, we must check if each *3D* head mesh vertex is inside some Candide triangle. This is also done in $O(Nn)$ operations. For the case when a vertex is not inside any Candide triangle a new distance must be computed. This step could need in the worst case $O(Nn)$ operations. Finally the computation of $\vec{\mu}$ also takes $O(Nn)$ in the worst case, and then the complete algorithm $O(N(n+1))$.

IV. Results

We present some animation results in Figure 3, where we test one of the main difficulties of the interpolation: as this kind of movements almost always involve several head points outside all Candide triangles, the implementation considerations we mention must be taken into account. Despite this, lips and jaw are naturally deformed in this example, and the overall animation seems quite realistic.

In the example we have shown, we have used the same data to animate two very different models, *JuanCarlos* [11] and *Alice*[6]. The animation data corresponds to the visual tracking during speech for a set of phonemes with the method proposed by [11]. The adaptation of the Candide3 model with our software takes around 5 minutes, including manual corrections, automatic projection and interpolation. Animation is achieved in real-time (more that $150\,fps$), even real-time recordings to secondary memory are possible with rates of around $25-30\,fps$ in a 2Ghz dual-core machine. For the animation we use the Ogre *3D* Engine [3], and for video capture and compression the OpenCV library [15]. We publish also two example videos http://wwww.loria.fr/~cerdavim/files/face for the JuanCarlos and Alice model with the same animation sequence.

V. Conclusions

The proposed interpolation algorithm is able to animate different face models maintaining the realism of the movements. We have shown that a quick adjustment of a generic face model (Candide3) is sufficient to provide satisfactory face animation using automatic data retrieved from video tracking.

Future work will include the evaluation of the perceptive quality of the animation using a group of human subjects. We are also considering to modify the interpolation method to include not only both rotation and translation, but also elastic deformations. In terms of implementation we will consider data structures more suited to the geometrical operations we perform to improve the performance of our algorithms.

Acknowledgments

This research work was supported in part by the Regional Programme for Scientific Cooperation STIC-AMSud: Project Bavi (09STIC06).

Figure 3. The same sequence with two models Alice (top) and JuanCarlos (bottom). From left to right: static, mouth open and mouth pronouncing "u" poses.

REFERENCES

[1] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231 – 268, 2001.

[2] N. Ersotelos and F. Dong, "Building highly realistic facial modeling and animation: a survey," *The Visual Computer*, vol. 24, no. 1, pp. 13–30, 2008.

[3] G. Junker, *Pro OGRE 3D Programming (Pro)*. Berkely, CA, USA: Apress, 2006.

[4] L. Forest, F. Padilla, S. Martinez, J. Demongeot, and J. S. Martin, "Modelling of auxin transport affected by gravity and differential radial growth," *Journal of Theoretical Biology*, vol. 241, no. 2, pp. 241–251, 2006.

[5] R. Medina, N. Hitschfeld, L. Forest, F. Padilla, S. Martinez, and J. S. Martin, "Geometric modelling fo tree stem deformation," in *5rd Symposium on Trends in Unstructured Mesh Generation, 7th World Congress on Computational Mechanics*, Los Angeles, California, 16-22 July 2006, p. Abstract Number: 342".

[6] K. Balci, "Xface: Mpeg-4 based open source toolkit for 3d facial animation," in *AVI '04: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA: ACM, 2004, pp. 399–402.

[7] S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann, "Feature point based mesh deformation applied to mpeg-4 facial animation," in *DEFORM '00/AVATARS '00: Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2001, pp. 24–34.

[8] N. Kojekine, V. Savchenko, M. Senin, and I. Hagiwara, "Real-time 3d deformations by means of compactly supported radial basis functions," in *Short papers proceedings of Eurographics*, 2002, pp. 35–43.

[9] S. Jakobsson and O. Amoignon, "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization," *Computers and Fluids*, vol. 36, no. 6, pp. 1119 – 1136, 2007.

[10] S. Garchery, R. boulic, T. Caping, and P. Kalra, "Standards for virtual humans," in *Handbook of Virtual Humans*, 1st ed., N. Magnenat-Thalmann and D. Thalmann, Eds. John Wiley & Sons, 2004, pp. 373–391.

[11] L. Terissi and J. Gómez, "3d head pose and facial expression tracking using a single camera," *Journal of Universal Computer Science*, vol. 16, no. 6, pp. 903–920, 2010.

[12] L. D. Terissi and J. C. Gomez, "Hmm inversion with full and diagonal covariance matrices for audio-to-visual conversion," in *SIGMAP*, P. A. A. Assunção and S. M. M. de Faria, Eds. INSTICC Press, 2008, pp. 168–173.

[13] J. Ahlberg, "Candide-3- an updated parameterised face," Department of Electrical Engineering, Linkoping University, Tech. Rep., 2001.

[14] S. Fortune, "Voronoi diagrams and delunay triangulations," *Handbook of discrete and computational geometry*, pp. 377–388, 1997.

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O'Reilly Media, Inc., October 2008.